Übungen zum Kapitel 4

NumPy-Grundlagen

armin.baenziger@zhaw.ch, 13. Januar 2020

```
In [1]: %autosave 0
```

```
Autosave disabled
```

Bevor wir beginnen, erstellen wir 3 Listen:

```
In [2]: list1 = [1, -2, 3, 0, 5, 2]
        list2 = [2, 5, 5, 1, -1, 0]
        list3 = [True, True, False, True, False, False]
```

(A.1) Laden Sie NumPy mit der üblichen Abkürzung.

```
In [3]: import numpy as np
```

(A.2) Erstellen Sie aus den drei Listen `list1`, `list2` und `list3` je einen NumPy-Array (Ndarray) mit den Namen `arr1`, `arr2`, `arr3`.

```
In [4]: arr1 = np.array(list1)
        arr2 = np.array(list2)
        arr3 = np.array(list3)
```

(A.3) Erstellen Sie einen Ndarray, bei dem jedes Element des Arrays das Doppelte von `arr1` ist.

```
In [5]: arr1 * 2
```

```
Out[5]: array([ 2, -4,  6,  0, 10,  4])
```

(A.4) Erstellen Sie einen (2, 6)-Array `arr2x6`, bei dem die zwei *Zeilen* den beiden Listen `list1` und `list2` entsprechen.

```
In [6]: arr2x6 = np.array([list1, list2])
        arr2x6
```

```
Out[6]: array([[ 1, -2,  3,  0,  5,  2],
               [ 2,  5,  5,  1, -1,  0]])
```

(A.5) Erstellen Sie den (6, 2)-Array `arr6x2`, bei dem die zwei *Spalten* den beiden Listen `list1` und `list2` entsprechen. Hinweis: `arr2x6` transponieren.

In [7]:
```
arr6x2 = arr2x6.T   # oder: arr2x6.transpose()
arr6x2
```

Out[7]:
```
array([[ 1,  2],
       [-2,  5],
       [ 3,  5],
       [ 0,  1],
       [ 5, -1],
       [ 2,  0]])
```

(A.6) Erstellen Sie einen (3, 4)-Array mit lauter Nullen. Hinweis: Funktion `np.zeros`

In [8]:
```
np.zeros((3, 4))
```

Out[8]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

(A.7) Erstellen Sie mit der Methode `reshape` einen (3, 2)-Array aus den Werten von `arr1` .

In [9]:
```
arr1.reshape(3, 2)
```

Out[9]:
```
array([[ 1, -2],
       [ 3,  0],
       [ 5,  2]])
```

(A.8) Erstellen Sie den (6, 3)-Array mit Name `arr6x3` , der sich aus fortlaufenden Zahlen (0, 1, .., 17) zusammensetzt. Verwenden Sie hierzu die Funktion `np.arange` in Kombination mit der Methode `reshape` .

In [10]:
```
arr6x3 = np.arange(18).reshape(6, 3)
arr6x3
```

Out[10]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11],
       [12, 13, 14],
       [15, 16, 17]])
```

(A.9) Erstellen Sie den (4, 3)-Array mit Name `arr4x3` , der sich aus standardnormalverteilten Zufallszahlen zusammensetzt. Verwenden Sie hierzu die Funktion `random.randn` .

In [11]:
```
np.random.seed(333)
arr4x3 = np.random.randn(4,3)
arr4x3
```

Out[11]:
```
array([[ 1.71718429,  0.32469333, -0.4745434 ],
       [-0.1379183 ,  0.52493957, -0.35783583],
       [-0.72820492, -0.64208645, -0.31704766],
       [-1.21900143, -0.05198034,  0.69402731]])
```

(B.1) Slicen Sie die ersten 3 Elemente aus `arr1` .

In [12]:
```
arr1[:3]
```

Out[12]:
```
array([ 1, -2,  3])
```

(B.2) Weisen Sie den ersten 2 Elementen von `arr1` den Wert 99 zu.

In [13]:
```
arr1[:2] = 99
arr1
```

Out[13]:
```
array([99, 99,  3,  0,  5,  2])
```

(B.3) Slicen Sie die ersten zwei Zeilen aus `arr6x3` .

In [14]:
```
arr6x3[:2]
```

Out[14]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

(B.4) Slicen Sie die ersten zwei Spalten aus `arr6x3` .

In [15]:
```
arr6x3[:,:2]
```

Out[15]:
```
array([[ 0,  1],
       [ 3,  4],
       [ 6,  7],
       [ 9, 10],
       [12, 13],
       [15, 16]])
```

(B.5) Erstellen Sie einen Array mit der ersten und dritten Zeile von `arr6x3` .

In [16]:
```
arr6x3[[0, 2]]
```

Out[16]:
```
array([[0, 1, 2],
       [6, 7, 8]])
```

(B.6) Ziehen Sie diejenigen Werte aus `arr1` , bei denen in `arr3` an der entsprechenden Position `True` steht.

In [17]:
```
arr1[arr3]
```

Out[17]:
```
array([99, 99,  0])
```

(B.7) Ziehen Sie diejenigen Werte aus `arr2` , bei denen in `arr1` an der entsprechenden Position `99` steht.

In [18]:
```
arr2[arr1 == 99]
```

Out[18]:
```
array([2, 5])
```

(B.8) Ziehen Sie diejenigen Werte aus `arr2` , bei denen in `arr1` an der entsprechenden Position `0` oder `99` steht.

In [19]:
```
arr2[(arr1 == 0) | (arr1 == 99)]
```

Out[19]:
```
array([2, 5, 1])
```

(B.9) Ersetzten Sie *alle* negativen Werte in `arr4x3` durch 0.

In [20]:
```
arr4x3[arr4x3 < 0] = 0
arr4x3
```

Out[20]:
```
array([[1.71718429, 0.32469333, 0.        ],
       [0.        , 0.52493957, 0.        ],
       [0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.69402731]])
```

(C.1) Erstellen Sie einen (4x5)-Array mit uniformverteilten (stetigen) Zufallsvariablen im Bereich `[0, 1)` (Funktion `np.random.rand`). Nennen Sie den Array `data`.

In [21]:
```
np.random.seed(123)
data = np.random.rand(4, 5)
data
```

Out[21]:
```
array([[0.69646919, 0.28613933, 0.22685145, 0.55131477, 0.71946897],
       [0.42310646, 0.9807642 , 0.68482974, 0.4809319 , 0.39211752],
       [0.34317802, 0.72904971, 0.43857224, 0.0596779 , 0.39804426],
       [0.73799541, 0.18249173, 0.17545176, 0.53155137, 0.53182759]])
```

(C.2) Berechnen Sie die Summe *aller* Werte in `data`.

In [22]:
```
data.sum()  # Der Wert ist sehr nahe am Erwartungswert von 20*0.5 = 10.
```

Out[22]:
```
9.569833503179613
```

(C.3) Berechnen Sie pro Spalte den Mittelwert (insgesamt 5) mit der Methode `mean` und dem Argument `axis=0`.

In [23]:
```
data.mean(axis=0)
```

Out[23]:
```
array([0.55018727, 0.54461124, 0.3814263 , 0.40586899, 0.51036458])
```

(C.4) Erstellen Sie eine Liste, welche das Element von `list1` enthält, falls das Element in `list3` an der entsprechenden Position `True` ist. Ansonsten wird das Element von `list2` genommen. Lösen Sie die Aufgabe mit der Funktion `np.where`.

In [24]:
```
list(np.where(list3, list1, list2))
```

Out[24]:
```
[1, -2, 5, 0, -1, 0]
```

Ende der Übung